| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 1 | BRS | L1 | 46 | appointment near3 schedule | EPO; JPO; DERWENT | 2004/11/04 13:12 |
| 2 | BRS | L2 | 1 | (appointment near3 schedule) and (book or reserve) | EPO; JPO; DERWENT | 2004/11/04 13:13 |
| 3 | BRS | L3 | 3 | pool near string | EPO; JPO; DERWENT | 2004/11/04 13:13 |
| 4 | BRS | L4 | 81 | linked near string | EPO; JPO; DERWENT | 2004/11/04 13:14 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 1 | IS&R | L1 | 2 | (("5632032") or ("5826081")).PN. | USPAT | 2004/11/04 10:46 |
| 2 | BRS | L2 | 13 | pool near string | USPAT | 2004/11/04 11:04 |
| 3 | BRS | L3 | 227 | appointment near3 schedule | USPAT | 2004/11/04 11:04 |
| 4 | BRS | L4 | 68 | 3 and (book or reserve) | USPAT | 2004/11/04 11:04 |
| 5 | BRS | L5 | 17 | 4 and string | USPAT | 2004/11/04 11:32 |
| 6 | BRS | L6 | 98 | linked near string | USPAT | 2004/11/04 11:32 |

```
### Status: Path 1 of [Dialog Information Services via Modem]

### Status: Initializing TCP/IP using (UseTelnetProto 1 ServiceID pto-dialog)
Trying 31060000009998...Open

DIALOG INFORMATION SERVICES
PLEASE LOGON:
 ******** HHHHHHHH SSSSSSSS?
### Status: Signing onto Dialog
 ********
ENTER PASSWORD:
 ******** HHHHHHHH SSSSSSSS? ********
Password incorrect
### Status: Incorrect Account Password.

### Status: Incorrect Account Password.

### Status: Path 1 of [Dialog Information Services via Modem]

### Status: Initializing TCP/IP using (UseTelnetProto 1 ServiceID pto-dialog)
Trying 31060000009998...Open

DIALOG INFORMATION SERVICES
PLEASE LOGON:
 ******** HHHHHHHH SSSSSSSS?
### Status: Signing onto Dialog
 ********
ENTER PASSWORD:
 ******** HHHHHHHH SSSSSSSS? ********
Welcome to DIALOG
### Status: Connected


Dialog level 04.18.01D

Last logoff:  03nov04 06:58:33
Logon file405 04nov04 13:45:57
           *** ANNOUNCEMENT ***
                    ***
--Connect Time joins DialUnits as pricing options on Dialog.
See HELP CONNECT for information.
                    ***
--SourceOne patents are now delivered to your email inbox
as PDF replacing TIFF delivery. See HELP SOURCE1 for more
information.
                    ***
--Important Notice to Freelance Authors--
See HELP FREELANCE for more information
                    ***
NEW FILES RELEASED
***Beilstein Abstracts (File 393)
***Beilstein Facts (File 390)
***Beilstein Reactions (File 391)
***F-D-C Gold/Silver Sheet (File 184)
***BIOSIS Toxicology (File 157)
***IPA Toxicology (File 153)
                    ***
UPDATING RESUMED
                    ***
RELOADED
***Toxfile (File 156)

REMOVED
***Textile Technology Digest (File 119)
                    ***

     >>> Enter BEGIN HOMEBASE for Dialog Announcements <<<
     >>>    of new databases, price changes, etc.      <<<
```

```
                    ****
COREABS is set ON as an alias for 77,35,593,65,2,233,99,473,474,475.
COREFULL is set ON as an alias for 9,15,16,20,148,160,275,476,610,613,621,623,624,636,8
10,813.
SOFTFULL is set ON as an alias for 278,634,256.
EUROFULL is set ON as an alias for 348,349.
JAPOABS is set ON as an alias for 347.
HEALTHFULL is set ON as an alias for 442,149,43,444.
HEALTHABS is set ON as an alias for 5,73,151,155,34,434.
DRUGFULL is set ON as an alias for 455,129,130.
DRUGABS is set ON as an alias for 74,42.
INSURANCEFULL is set ON as an alias for 625,637.
INSURANCEABS is set ON as an alias for 169.
TRANSPORTFULL is set ON as an alias for 80,637.
TRANSPORTABS is set ON as an alias for 108,6,63.
ADVERTISINGFULL is set ON as an alias for 635,570,PAPERSMJ,PAPERSEU.
INVENTORYABS is set ON as an alias for 8,14,94,6,34,434,7.
BANKINGFULL is set ON as an alias for 625,268,626,267.
BANKINGABS is set ON as an alias for 139.
HEALTHALL is set ON as an alias for COREFULL,COREABS,HEALTHFULL,HEALTHABS.
INSURANCEALL is set ON as an alias for COREFULL,COREABS,INSURANCEFULL,INSURANCEABS.
RESERVATIONALL is set ON as an alias for COREFULL, COREABS.
OPERATIONSALL is set ON as an alias for COREFULL,COREABS,INVENTORYABS.
TRANSPORTALL is set ON as an alias for COREFULL,COREABS,TRANSPORTFULL,TRANSPORTABS.
ADVERTISINGALL is set ON as an alias for COREFULL,COREABS,ADVERTISINGFULL.
SHOPPINGALL is set ON as an alias for COREFULL,COREABS,ADVERTISINGALL,47.
INVENTORYALL is set ON as an alias for COREFULL,COREABS,INVENTORYFULL.
BANKINGALL is set ON as an alias for COREFULL,COREABS,BANKINGFULL,BANKINGABS.
PORTFOLIOALL is set ON as an alias for COREFULL,COREABS,BANKINGALL.
TRADINGALL is set ON as an alias for COREFULL,COREABS,BANKINGALL.
CREDITALL is set ON as an alias for COREFULL,COREABS,BANKINGALL.
FUNDSALL is set ON as an alias for COREFULL,COREABS,BANKINGALL,608.
* * *
SYSTEM:HOME
Cost is in DialUnits
Menu System II: D2 version 1.7.9 term=ASCII
                    *** DIALOG HOMEBASE(SM) Main Menu ***

 Information:
  1.  Announcements (new files, reloads, etc.)
  2.  Database, Rates, & Command Descriptions
  3.  Help in Choosing Databases for Your Topic
  4.  Customer Services (telephone assistance, training, seminars, etc.)
  5.  Product Descriptions

 Connections:
  6.  DIALOG(R) Document Delivery
  7.  Data Star(R)

    (c) 2003 Dialog, a Thomson business.      All rights reserved.

      /H = Help            /L = Logoff          /NOMENU = Command Mode


Enter an option number to view information or to connect to an online
 service.  Enter a BEGIN command plus a file number to search a database
(e.g., B1 for ERIC).
?b corefull, coreabs

>>>          77 does not exist
>>>1 of the specified files is not available
       04nov04 13:46:17 User242933 Session D189.1
            $0.00    0.264 DialUnits FileHomeBase
     $0.00  Estimated cost FileHomeBase
     $0.08  TELNET
     $0.08  Estimated cost this search
     $0.08  Estimated total session cost   0.264 DialUnits
```

SYSTEM:OS - DIALOG OneSearch
File 9:Business & Industry(R) Jul/1994-2004/Nov 03
(c) 2004 The Gale Group
File 15:ABI/Inform(R) 1971-2004/Nov 04
(c) 2004 ProQuest Info&Learning
***File 15: Alert feature enhanced for multiple files, duplicate**
removal, customized scheduling. See HELP ALERT.
File 16:Gale Group PROMT(R) 1990-2004/Nov 04
(c) 2004 The Gale Group
***File 16: Alert feature enhanced for multiple files, duplicate**
removal, customized scheduling. See HELP ALERT.
File 20:Dialog Global Reporter 1997-2004/Nov 04
(c) 2004 The Dialog Corp.
File 148:Gale Group Trade & Industry DB 1976-2004/Nov 04
(c)2004 The Gale Group
***File 148: Alert feature enhanced for multiple files, duplicate**
removal, customized scheduling. See HELP ALERT.
File 160:Gale Group PROMT(R) 1972-1989
(c) 1999 The Gale Group
File 275:Gale Group Computer DB(TM) 1983-2004/Nov 04
(c) 2004 The Gale Group
File 476:Financial Times Fulltext 1982-2004/Nov 04
(c) 2004 Financial Times Ltd
File 610:Business Wire 1999-2004/Nov 01
(c) 2004 Business Wire.
***File 610: File 610 now contains data from 3/99 forward.**
Archive data (1986-2/99) is available in File 810.
File 613:PR Newswire 1999-2004/Nov 04
(c) 2004 PR Newswire Association Inc
***File 613: File 613 now contains data from 5/99 forward.**
Archive data (1987-4/99) is available in File 813.
File 621:Gale Group New Prod.Annou.(R) 1985-2004/Nov 04
(c) 2004 The Gale Group
File 623:Business Week 1985-2004/Nov 02
(c) 2004 The McGraw-Hill Companies Inc
File 624:McGraw-Hill Publications 1985-2004/Nov 02
(c) 2004 McGraw-Hill Co. Inc
***File 624: Homeland Security & Defense and 9 Platt energy journals added**
Please see HELP NEWS624 for more
File 636:Gale Group Newsletter DB(TM) 1987-2004/Nov 04
(c) 2004 The Gale Group
File 810:Business Wire 1986-1999/Feb 28
(c) 1999 Business Wire
File 813:PR Newswire 1987-1999/Apr 30
(c) 1999 PR Newswire Association Inc
File 35:Dissertation Abs Online 1861-2004/Oct
(c) 2004 ProQuest Info&Learning
File 593:KOMPASS Central/Eastern Europe 2004/Jul
(c) 2004 KOMPASS Intl.
File 65:Inside Conferences 1993-2004/Oct W5
(c) 2004 BLDSC all rts. reserv.
File 2:INSPEC 1969-2004/Oct W4
(c) 2004 Institution of Electrical Engineers
***File 2: Alert feature enhanced for multiple files, duplicates**
removal, customized scheduling. See HELP ALERT.
File 233:Internet & Personal Comp. Abs. 1981-2003/Sep
(c) 2003 EBSCO Pub.
***File 233: File 233 is closed (no longer updating).**
File 99:Wilson Appl. Sci & Tech Abs 1983-2004/Sep
(c) 2004 The HW Wilson Co.
File 473:FINANCIAL TIMES ABSTRACTS 1998-2001/APR 02
(c) 2001 THE NEW YORK TIMES
***File 473: This file will not update after March 31, 2001.**
It will remain on Dialog as a closed file.
File 474:New York Times Abs 1969-2004/Nov 03
(c) 2004 The New York Times
File 475:Wall Street Journal Abs 1973-2004/Nov 03
(c) 2004 The New York Times

```
      Set  Items  Description
      ---  -----  -----------
?s (pool (w) string)
          800107  POOL
          366785  STRING
      S1       0  (POOL (W) STRING)
?s (linked (w) string)
         1174052  LINKED
          366785  STRING
      S2       9  (LINKED (W) STRING)
?type s2/3,ab/all
>>>No matching display code(s) found in file(s): 65, 593, 623-624, 810, 813
```

Considered abstracts ID

**2/3,AB/1 (Item 1 from file: 16)**
DIALOG(R)File 16:Gale Group PROMT(R)
(c) 2004 The Gale Group. All rts. reserv.

11271515 Supplier Number: 118173581
**Sizing up the mighty mote.(Net Results)**
Lowe, Jonathan W.
Geospatial Solutions, v14, n5, p42
May, 2004
Language: English Record Type: Fulltext
Document Type: Magazine/Journal; General
Word Count: 2856

**2/3,AB/2 (Item 2 from file: 16)**
DIALOG(R)File 16:Gale Group PROMT(R)
(c) 2004 The Gale Group. All rts. reserv.

03066377 Supplier Number: 44174129
**Tumour tissue treatment**
Pharmaceutical Business News, pN/A
Oct 18, 1993
Language: English Record Type: Fulltext
Document Type: Newsletter; Trade
Word Count: 557

**2/3,AB/3 (Item 1 from file: 20)**
DIALOG(R)File 20:Dialog Global Reporter
(c) 2004 The Dialog Corp. All rts. reserv.

11109760
**Arts review: Classical - Philharmonia/Conta Royal Festival Hall London**
INDEPENDENT
May 19, 2000
JOURNAL CODE: FIND LANGUAGE: English RECORD TYPE: FULLTEXT
WORD COUNT: 527

TCHAIKOVSKY NIGHTS have changed from the cheap and cheerful crowd-pleasers that used to fill the Royal Albert Hall on Sundays. Partly that's because the repertoire of pop classics has diversified, but it also shows that British taste-makers have grown up. No longer afraid to be seen liking music that oozed "self-pity" and lacked "backbone", they happily present Tchaikovsky in themed series, like other great composers, if they are not actively promoting him as a gay icon that is.

So the Philharmonia can continue its Russian spring, which Valery Gergiev's conducting vividly started last week, by inviting Mikhail Pletnev and the Romanian-born conductor Vladimir Conta to play all the Tchaikovsky piano concertos around England for a week. Years of living with these Russian national treasures have refined Pletnev's performances. They are subtle, sophisticated, powerful when necessary and always satisfying. They include one piece of buried treasure, the rare Concert Fantasy.

**2/3,AB/4 (Item 2 from file: 20)**
DIALOG(R)File 20:Dialog Global Reporter

11098887
**Classical: Philharmonia/Conta Royal Festival Hall London**
INDEPENDENT
May 19, 2000
JOURNAL CODE: FIND LANGUAGE: English RECORD TYPE: FULLTEXT
WORD COUNT: 527

TCHAIKOVSKY NIGHTS have changed from the cheap and cheerful crowd-pleasers that used to fill the Royal Albert Hall on Sundays. Partly that's because the repertoire of pop classics has diversified, but it also shows that British taste-makers have grown up. No longer afraid to be seen liking music that oozed "self-pity" and lacked "backbone", they happily present Tchaikovsky in themed series, like other great composers, if they are not actively promoting him as a gay icon that is.

So the Philharmonia can continue its Russian spring, which Valery Gergiev's conducting vividly started last week, by inviting Mikhail Pletnev and the Romanian-born conductor Vladimir Conta to play all the Tchaikovsky piano concertos around England for a week. Years of living with these Russian national treasures have refined Pletnev's performances. They are subtle, sophisticated, powerful when necessary and always satisfying. They include one piece of buried treasure, the rare Concert Fantasy.

**2/3,AB/5 (Item 1 from file: 148)**
DIALOG(R)File 148:Gale Group Trade & Industry DB
(c)2004 The Gale Group. All rts. reserv.

12525423 SUPPLIER NUMBER: 64826262 (USE FORMAT 7 OR 9 FOR FULL TEXT)
**Life before television; legends with a purpose ...**
National Post, 2, 239, B8
July 21, 2000
ISSN: 1493-4779 LANGUAGE: English RECORD TYPE: Fulltext
WORD COUNT: 382 LINE COUNT: 00032

**2/3,AB/6 (Item 2 from file: 148)**
DIALOG(R)File 148:Gale Group Trade & Industry DB
(c)2004 The Gale Group. All rts. reserv.

05539180 SUPPLIER NUMBER: 11639669 (USE FORMAT 7 OR 9 FOR FULL TEXT)
**Potential for CNG vehicles depends on strong infrastructure. (compressed natural gas) (Brief Article)**
Drummond, Jim
Oil Daily, n9885, p1(2)
Dec 9, 1991
DOCUMENT TYPE: Brief Article ISSN: 0030-1434 LANGUAGE: ENGLISH
RECORD TYPE: FULLTEXT
WORD COUNT: 555 LINE COUNT: 00044

**2/3,AB/7 (Item 1 from file: 636)**
DIALOG(R)File 636:Gale Group Newsletter DB(TM)
(c) 2004 The Gale Group. All rts. reserv.

02203382 Supplier Number: 44174129
**Tumour tissue treatment**
Pharmaceutical Business News, pN/A
Oct 18, 1993
Language: English Record Type: Fulltext
Document Type: Newsletter; Trade
Word Count: 557

**2/3,AB/8 (Item 1 from file: 2)**
DIALOG(R)File 2:INSPEC
(c) 2004 Institution of Electrical Engineers. All rts. reserv.

5020891 INSPEC Abstract Number: A9517-1117-009
**Title: Non-zero helicity generalization of the Nielsen-Olesen string solution**
Author(s): Owczarek, R.
Author Affiliation: Inst. of Fundamental Technol. Res., Acad. of Sci., Warsaw, Poland
Journal: Reports on Mathematical Physics vol.34, no.3 p.305-10
Publication Date: Dec. 1994 Country of Publication: UK
CODEN: RMHPBE ISSN: 0034-4877
Language: English
Abstract: A solution generalizing the Nielsen-Olesen string solution is found. Its existence in the stationary ansatz is proved and non-triviality of the helicity integral is shown. Monotonicity properties of the solution are discussed. Asymptotic behaviour in regions far away from the core of the string and near the core are found. Numerical solution is presented. Finally, importance for knotted and **linked string** structures in superfluids and superconductors is suggested.
Subfile: A


**2/3,AB/9 (Item 2 from file: 2)**
DIALOG(R)File 2:INSPEC


4977282 INSPEC Abstract Number: A9514-1117-003
**Title: Zero modes on linked strings**
Author(s): Garriga, J.; Vachaspati, T.
Author Affiliation: Univ. Autonoma de Barcelona, Spain
Journal: Nuclear Physics B vol.B438, no.1-2 p.161-81
Publication Date: 27 March 1995 Country of Publication: Netherlands
CODEN: NUPBBO ISSN: 0550-3213
U.S. Copyright Clearance Center Code: 0550-3213/95/$09.50
Language: English
Abstract: We study linked loops of strings in the presence of bosonic condensates and fermionic zero modes on the strings. We find that the strings necessarily carry a current if the bosons have an Aharanov-Bohm interaction with the string. The fermionic case is analyzed in the context of the standard model where there are lepton and quark zero modes on Z-strings. Here we find that the fermionic ground state in the **linked string** background is lower than the ground state when the loops are unlinked but otherwise identical. As in the bosonic case, the Z-strings carry a non-vanishing electric current in the ground state. The baryon number of the linked configuration is found to agree with previous indirect results. We also evaluate the angular momentum, electromagnetic charge and baryonic three current on the linked Z-string configuration. Finally we point out a possible gravitational analogue of the **linked string** system.
Subfile: A

?s (memory (n) management) and (string or strings)
Processing
Processed 10 of 25 files ...
Completed processing all files
1404945 MEMORY
15014412 MANAGEMENT
30357 MEMORY(N)MANAGEMENT
366785 STRING
126117 STRINGS
S3 1121 (MEMORY (N) MANAGEMENT) AND (STRING OR STRINGS)
?s s3 and (pool (n5) (string or strings))
1121 S3
800107 POOL
366785 STRING
126117 STRINGS
168 POOL(5N)(STRING OR STRINGS)
S4 2 S3 AND (POOL (N5) (STRING OR STRINGS))
?type s4/3,ab/all
>>>No matching display code(s) found in file(s): 65, 593, 623-624, 810, 813

**4/3,AB/1 (Item 1 from file: 275)**
DIALOG(R)File 275:Gale Group Computer DB(TM)


01447129 SUPPLIER NUMBER: 11213229 (USE FORMAT 7 OR 9 FOR FULL TEXT)
**Windows 3.1 - hello to TrueType, OLE, and easier DDE; farewell to real mode. (object linking and embedding, dynamic data exchange)**
Petzold, Charles
Microsoft Systems Journal, v6, n5, p17(10)
Sept, 1991
ISSN: 0889-9932 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 7472 LINE COUNT: 00577

ABSTRACT: Microsoft Windows 3.1 will offer extensive applications programming interface (API) enhancements including Object Linking and Embedding (OLE) and TrueType outline font technology. Windows 3.1 will also not run in real mode which will allow developers to write programs without the need for frequently locking and unlocking global memory blocks. TrueType will be available to both Windows users and programmers and is a potential industry standard since it is an open technology, allowing any font manufacturers to develop fonts for the format. Problems with Dynamic Data Exchange (DDE) are addressed by the Dynamic Data Exchange Management Library (DDEML), an API layer built on top of the message-based DDE. Windows 3.1 provides 64 new function calls for OLE and is the program's most complex enhancement, providing a format for compound documents. Further details are presented.

**4/3,AB/2 (Item 2 from file: 275)**
DIALOG(R)File 275:Gale Group Computer DB(TM)


01346253 SUPPLIER NUMBER: 08102586 (USE FORMAT 7 OR 9 FOR FULL TEXT)
**FoxPro! (Software Review) (includes related article on FoxPro's memo fields, and a related article on changes to data files fromFoxBASE to FoxPro) (evaluation)**
Hawkins, John L.
Data Based Advisor, v8, n2, p70(15)
Feb, 1990
DOCUMENT TYPE: evaluation ISSN: 0740-5200 LANGUAGE: ENGLISH
RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 8318 LINE COUNT: 00652

ABSTRACT: FoxPro is an outstanding addition to the data base management system market. It features a new interface, good documentation and on-line help, an internal editor that provides sufficient program writing capability for many users, a report writer that allows creation of complex reports and labels without additional programming, and new commands and functions, including CURDIR, which returns the current DOS directory on any drive. Speed is estimated at from one to as much as 16 times greater than other data base management systems. FoxPro's biggest drawback is its memory consumption. Connectivity will be provided by FoxPro/LAN, due to be released early in 1990. FoxPro is $795 for a single-user version, $1,095 for FoxPro/LAN, and an additional $500 for an unlimited runtime version. Upgrades from FoxBASE+ to FoxPro are $195, to FoxPro/LAN from FoxBASE+/LAN $250, and from FoxBASE+ Runtime to FoxPro Runtime $50.
?type s4/3,9/all

**4/9/1 (Item 1 from file: 275)**
DIALOG(R)File 275:Gale Group Computer DB(TM)


01447129 SUPPLIER NUMBER: 11213229 (THIS IS THE FULL TEXT)
**Windows 3.1 - hello to TrueType, OLE, and easier DDE; farewell to real mode. (object linking and embedding, dynamic data exchange)**
Petzold, Charles
Microsoft Systems Journal, v6, n5, p17(10)
Sept, 1991

ISSN: 0889-9932 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 7472 LINE COUNT: 00577

ABSTRACT: Microsoft Windows 3.1 will offer extensive applications programming interface (API) enhancements including Object Linking and Embedding (OLE) and TrueType outline font technology. Windows 3.1 will also not run in real mode which will allow developers to write programs without the need for frequently locking and unlocking global memory blocks. TrueType will be available to both Windows users and programmers and is a potential industry standard since it is an open technology, allowing any font manufacturers to develop fonts for the format. Problems with Dynamic Data Exchange (DDE) are addressed by the Dynamic Data Exchange Management Library (DDEML), an API layer built on top of the message-based DDE. Windows 3.1 provides 64 new function calls for OLE and is the program's most complex enhancement, providing a format for compound documents. Further details are presented.

TEXT:

You may have heard that windows 3.1 is basically' Windows 3.0 plus True TYPO plus some minor fixes.' If so, you heard wrong. WindOws 3.1 actually represents the most extensive API enhancement to Windows in its history. You'll laugh, you'll cry, you'll experience header shock, I but you certainly won't be bored.

There's an old saying in this young industry that you should never buy or develop for operating system software with a "point zero" version number-you should wait for the "point one" or "point zero one" version so all the bugs (or at least the more blatant ones) will be fixed.

Microsoft did its best to alleviate point-zero anxiety with extensive beta testing on the Microsoft" Windows" graphical environment 3.0 and MS-DOS" 5 operating system. This approach seems to have worked: these products entered the market in generally good shape and millions of users grabbed them up.

But Redmond isn't stopping there. The move to protected mode-without a doubt the most significant enhancement to Windows' 3.0-was just the beginning. Protected mode makes available the memory to implement lots of other neat stuff, and an update was inevitable.

Yes, friends, just when you thought it was safe to take a vacation, here comes Windows 3.1.

You may have heard that Windows 3.1 is basically "Windows 3.0 plus TrueType" plus some minor fixes." If so, you heard wrong. Windows 3.1 actually represents the most extensive API enhancement to Windows in its six-year history. You'll laugh, you'll cry, you'll experience "header shock," but you certainly won't be bored.

As you can see from Figure 1, Windows 3.1 has a third more function calls than Windows 3.0. The Multimedia and Windows for Pens extensions bring the total up to over 1000. Whew ! (This discussion of Windows 3.1 is based on information distributed with the first beta release in June 1991, which is subject to change-Ed.)

Porting from Windows 2.x to 3.0 was easy for some and difficult for others. For many programs that followed the Windows 2.x **memory management** rules, upgrading to run under Windows 3.0 was not a major chore. Basically you had to make sure your program ran OK in protected mode, and that it didn't assume the system font was fixed-pitch.

When Windows 3.0 was released, everyone knew it was missing TrueType. TrueType is the Apple" outline font technology that promises to deliver real VATYSIWYG to the Macintosh" and Windows. Taking advantage of TrueType may require some changes to your code, as will be discussed shortly. Otherwise, porting to Windows 3. 1 should be much easier than porting from Windows 2.x to 3.0, unless you want to take advantage of some of the new features. That's up to you.

Some of the new function calls in Windows 3.1 have been desperately desired since Windows 1.0 (yes, we now have a standard File Open dialog box API), while others seem rather strange (was the absence of Lempel-Ziv data decompression algorithms really a big hole in Windows?).

Windows 3.1 is also the platform for OLE, which promises to be the most famous and infamous Windows TLA since DDE. (That's Object Linking and Embedding, Three Letter Acronym, and Dynamic Data Exchange.) Like DDE, OLE support will be a common item in feature comparison charts in Windows software reviews. So even if your application has nothing that can possibly

be used with OLE, you're going to have to put it in anyway.

One of the more interesting changes is that windows 3.1 will not run in real mode. As you know, more and more Windows applications refuse to run in real mode, and the vast majority of Windows users have the necessary hardware (a 286 processor or better with at least IMB of memory) for running Windows in standard mode. No real mode means that you can write Windows 3.1 programs without worrying about frequently locking and unlocking your global memory blocks. You can allocate a block, lock it to return a pointer, and keep the pointer around until you're ready to free the block. Then you can obtain the block's handle from GlobalHandle, unlock it, and free it.

What's in it for the User?

It was once common for Windows programmers to do all their work from the DOS2 command line using DOS applications, running Windows only when testing their code. But with products such as Borland(R) C++, Turbo Pascal" for Windows, the Microsoft Visual programming system, and the QuickC(R) graphical development system for Windows, the Windows programmer is now also becoming a Windows user.

Version 3.1 includes several enhancements to the Windows environment that users will experience. The most significant is a new File Manager. (Yes, I know-you just figured out how to use the old File Manager and now it's being replaced.)

In version 3.0, the File Manager is an MDI (Multiple Document Interface) application that displays one window with a directory tree and zero or more windows listing the files in a particular subdirectory (see Figure 2).

The first version of the File Manager has a big problem when switching the directory tree window among different disks, either local or on the network. After each switch, the File Manager has to rebuild the directory tree.

In Windows 3. 1, each child window in the File Manager is split in two. The left side displays a directory tree and the right side lists the files in a particular directory (see Figure 3).

You can create new tree-and-file windows even on multiple disks. The really great thing about this is that the File Manager remembers all the open directory windows. If you leave Windows and reenter, the File Manager reopens all the previously opened windows. This is great if you need to dive in and out of Windows frequently.

The Windows 3.1 File Manager shows significantly better performance than the 3.0 version. On my 500directory hard disk, the Windows 3.0 File Manager requires 35 seconds to construct the entire directory tree. The Windows 3.1 File Manager gets that down to 29 seconds. And I hear the improvement over a network is much more dramatic.)

A new program group called the Startup group has been added to the Program Manager. Add (or drag) a program to this group and the program is automatically run when you start up Windows. It's a whole lot easier than using the Run option in WIN.INI.

Windows 3.1 also supports screen savers, those wacky programs that do odd things to your screen when you haven't been typing or mousing for awhile. The screen saver protocol in Windows 3.1 is the same as in Windows with Multimedia Extensions.

Unacceptable API Errors

One of the best things about protected-mode Windows is the Unrecoverable Application Error (UAE). Protected mode lets Windows trap errant applications and (we hope) terminate them before they cause damage to the rest of the system.

One of the worst things about protected-mode Windows is also the Unrecoverable Application Error. It gives you absolutely no information about what caused the problem-the program? a dynamic-link library? a bad function call? a bad function?-and no way to correct it. Even when the program is terminated, Windows may not clean up properly, or it may be left in an unstable state, or it may just simply crash.

The UAE has become the most hated "feature" of Windows 3.0 and is badly in need of fixing. Of course, that's easier said than done. The implementation of protected mode in Windows (even in 3. 1) does not take advantage of the 286 and 386 hardware properly to separate tasks. All Windows applications and Windows itself share the same Local Descriptor Table (LDT).

Microsoft's investigation shows that about 40 percent of UAEs occur

in applications themselves due to addressing bugs, while the other 60 percent result from passing invalid parameters to API calls. Of course, what constitutes an "invalid parameter" may be a matter of interpretation. My most recent struggle with mysterious UAEs in Windows 3.0 resulted from calling PolyPolygon with a final parameter of 1. (This bug has been fixed under 3. 1.) And let's face it-system API calls should check for valid parameters and be made as bulletproof as possible. Currently, a UAE is too often an Unacceptable API Error.

Microsoft has attempted to minimize UAEs under Windows 3.1 with better parameter checking. The UAE message box is also different. In Windows 3.0, it simply says

UNRECOVERABLE APPLICATION ERROR

Terminating current application.

Under Windows 3.1, the corresponding message box provides some additional information to help developers track down the problem. It reads something like this:

Application Error

BADAPP caused a General Protection Fault in module BADAPP.EXE at 0001:00AE.

Press Close. BADAPP will close.

I don't know if it will be included in the retail release of Windows 3.1, but a handy little program in the beta release called Dr. Watson provides more information when a UAE occurs. Dr. Watson creates a log file that includes the address of the fault, disassembled instructions leading up to the fault, the contents of the CPU registers, a stack dump, a list of the tasks and DLLs currently loaded under Windows, and some other internal information about the state of Windows.

In the space of a couple hours, Dr. Watson helped me track down the cause of a UAE in one of my programs that had been plaguing me for months. Check it out.

TrueType

The definition of word processor NWSIWYG has certainly changed over the past ten years. Would you believe that WordStar 3.x was once considered WYSIWYG because it displayed correct printer line and page breaks? Many word processors in the old days did not. Later on, Microsoft Word was considered WYSIWYG because it showed italic, underlining, and boldface attributes on the screen. (Never mind that all fonts were displayed at the same size!) Windows Write was later termed WYSIWYG because various font faces and sizes were displayed.

But despite the inclusion in Windows of several fonts and sizes, the video display of formatted documents never quite matched the printer. The problem was that Windows screen fonts were stored as bitmaps. These bitmap (or "raster") fonts were pegged to specific display resolutions and font sizes, and could not easily be manipulated. Raster fonts could be made larger in integral multiples by duplicating rows and columns of pixels, but this resulted in ugly stairsteps. The only alternative to the raster fonts were the "stroke" fonts. Since they were drawn using simple unfilled polylines, stroke fonts could be stretched and compressed. That's fine for a plotter but typographically absurd otherwise.

Meanwhile, printers have become more sophisticated. In particular, PostScript" has demonstrated the power of infinitely scalable outline fonts-fonts defined by filled outlines that can be scaled, skewed, and rotated. Programmers working with the Graphics Programming Interface (GPI) of OS/28 1.1 got a taste of the power of using outline fonts, as demonstrated in my article "OS/2 Graphics Programming Interface: An Introduction to Coordinate Spaces," MSJ (Vol. 3, No. 4).

The GPI outline fonts had one very serious flaw. The scaling was blind and did not make any adjustments for typographical correctness. For example, because of rounding errors, the two legs of a Helvetica(R) H might turn out to be different in width by one pixel. At normal font sizes on the screen, this inaccuracy led to virtually unreadable text.

Adjusting fonts for typographical correctness during scaling is called hinting. Windows did not get hinted scalable outline fonts until third-party utilities under Windows 3.0 such as Adobe Type Manager" were available.

Windows 3.1 introduces a new outline font technology called TrueType to Windows users and programmers. TrueType was developed initially by Apple for the Macintosh (it's implemented in their new System 7.0 operating system) then licensed to Microsoft. As the outline font technology included

with the two most popular graphical platforms for the desktop, TrueType has the potential to become an industry standard. TrueType is an open standard-any font manufacturer can develop fonts in the TrueType format. TrueType fonts can be used on any graphics output device supported by Windows.

Microsoft has provided TrueType fonts with Windows 3.1 (see Figure 4) which are equivalent to the 13 "standard" fonts familiar to PostScript users Courier, Helvetica, and Times Roman in normal, bold, italic, and bold italic versions, plus a Symbol font . However Microsoft has not licensed the Helvetica and Times Roman names. Instead, you'll have to start getting used to the names Arial(R) and Times New Roman(R) (Courier continues to be named Courier). These 13 TrueType fonts have the same metrics as the corresponding PostScript fonts.

Each of the 13 TrueType fonts are stored in a file with the extension TTF (TrueType Font). These files range in size from about 53KB to 70KB. It is likely that TrueType fonts from other font manufacturers will be about the same size. Although these are large files compared with some of the raster font files, keep in mind that a single TrueType file provides a font for all output devices, all display resolutions, and all font sizes.

Microsoft is recommending that Windows programs use TrueType exclusively for displaying text on the screen and printer.

However, a Windows user can disable TrueType or uninstall all TrueType fonts. Thus, to make your programs usable by everyone, you'll want to integrate TrueType-awareness into your programs while retaining compatibility with existing raster and stroke fonts.

If your program knows that all TrueType fonts are available, using them is very easy. You set the IfFaceName field of the LOGFONT structure to either Courier, Arial, Times New Roman, or Symbol. You set the IfHeight field to the desired font height in logical units. (Normally, this field indicates a height including internal leading.

This is equivalent to the desired line spacing rather than the point size. Use a negative value if you want the absolute value of lfHeight to represent the point size of the font.) You optionally set the lfWeight field to 700 (for bold) and the IfItalic, Ifunderline, or lfStrikeOut fields to 1. (You can also optionally set the lfWidth field to a nonzero value if you want a nonstandard width.) All other fields of the LOGFONT structure can be set to zero. Pass a pointer to the LOGFONT structure to CreateFontIndirect, and you're ready to select the font handle into a device context for displaying text.

When using TrueType fonts on the screen, the outlines are scaled to the desired height and then rasterized (that is, converted to bitmaps). Once these bitmaps are cached in memory, they are as fast as normal raster fonts. When using TrueType on a printer, GDI can either send the fonts to the printer in the form of bitmaps or filled polygons, or the device driver can convert TrueType font data to a downloadable printer font.

The TrueType italic fonts are truly italic, not just oblique versions of the normal fonts. This is evident, for example, by the difference in the lowcrease "a" and "f" in the Times New Roman and Times New Roman Italic fonts (see Figure 4).

Some of the previously unused ASCII codes between 80H and 9FH are now used for some common desktop publishing symbols (see Figure 5). The Symbol font is compatible with the same font in PostScript (see Figure 6).

Existing Windows applications that work with different fonts usually call EnumFonts and report fonts to the user for selection. Some of these applications will work, unmodified, with TrueType while others will not. Basically it depends on whether the application deliberately excludes stroke fonts by checking the zero bit of the nFontType parameter to the EnumFonts callback function. Applications that exclude stroke fonts will also exclude TrueType fonts. (This is the case with the JUSTIFY program in my book Programming Windows (Microsoft Press, 1990). I always thought the stroke fonts were feeble and I didn't want to both with them.)

Because the old raster fonts used the ANSI character set and the old stroke fonts used the OEM (that is, IBM-" PC) character set, some programs may assume that only fonts using the OEM character set are scalable. This is incorrect. Code that makes this assumption will have to be fixed to be compatible with TrueType.

A Windows application that includes the stroke fonts for user selection-and also gives the user the opportunity to select a precise font size for these fonts-is in excellent shape. Most likely, a program such as

this will pick up and use TrueType fonts without change.

You can identify a TrueType font in your code by a new SCALABLE_FONTTYPE flag in the nFontType parameter to the EnumFonts callback function. If the flag is 1, the font is a TrueType font. The encoding of the nFontType parameter for various screen and printer fonts is summarized in Figure 7.

There's a second method for identifying a TrueType font when calling GetTextMetrics. The bottom four bits of the lfPitchAndFamily field are encoded as shown in Figure 8. Bit 2 is the new TrueType bit.

Using TrueType fonts exclusively on the screen and printer is a simple matter of ignoring all fonts where the SCALABLE_FONTTYPE flag is not set. Exclusive use of TrueType solves some basic problems. First, the screen and printer fonts always match in both appearance and size. Second, you get the same character set on the screen and printer. Third, you begin approaching document compatibility with TrueType documents created on the Macintosh.

However, users who have invested in font cartridges and downloadable printer fonts might be irked that they are unable to use these fonts in your application. Moreover, some user may perversely disable TrueType or uninstall all TrueType fonts.

A better approach is to report all printer fonts to the user but encourage users to use TrueType. If you use EnumFonts to put the fonts in a list box, the TrueType font names appear as mixed uppcrease and lowercase. All other font names are in uppercase. This is the third method of identifying a TrueType font. It's a good visual difference if the user is aware of the distinction.

You'll probably want to write a dialog box to allow the user to choose fonts (listing the font names and standard sizes) and give the user an opportunity to specify a specific size for scalable fonts. Oh, you don't feel like writing this dialog box? I have some good news for you later in this article.)

One of the great things about outline fonts is that the outlines can be manipulated like any other type of graphical object. For example, a font can be easily rotated. However, when implementing TrueType in Windows 3. 1, Microsoft focused on performance and legibility rather than special effects. In the first beta release of Windows 3.1 available in June, the TrueType fonts did not respond to the lf Escapement or lfOrientation fields of the LOGFONT structure. This may change in the retail release.

Windows 3.1 still includes the old raster and stroke fonts. However, the raster fonts have been given some new names. The old Helv font is now called MS Sans Serif and the old Tms Rmn font is now MS Serif. If your program refers to these fonts by their old names in CreateFont or CreateFontIndirect calls, don't fret. The old names are aliased to the new names. However, if you call EnumFont and search for the old names, you'll have to make some changes to your code.

Several new function calls have been introduced to support TrueType. GetRasterizerCaps indicates whether TrueType is enabled and whether at least one TrueType font is installed. It could be that TrueType is enabled but all the TrueType fonts have been uninstalled by the user.

You've always been able to get individual character widths by calling GetCharWidth. With Windows 3. 1, you can call GetCharABCWidths to get character widths for TrueType fonts broken down into three segments. The A and C widths are the white space before and after the character. The B width is the width of the character itself. You can use this information to orient lines of text properly. For example, the A width of the first character in a line can be subtracted from the x coordinate passed to TextOut.

The A and C widths don't necessarily have to represent only white space. (In fact, they can be negative.) The A and C widths should be based on optical character margins rather than real character margins. For example, if one line of text begins with a capital E and the next line begins with a capital O, some of the left part of the O should be further left than the left of the E to produce the same visual margin. The A width of the 0 might reflect that by overlapping the character slightly.

The new GetOutlineTextMetrics is similar to GetTextMetrics except that it can only be used with TrueType fonts and it uses a structure of OUTLINETEXTMETRIC rather than TEXTMETRIC. This structure provides additional detailed information about the font.

EnumFontFamilies works similarly to the EnumFonts function. However,

for a specified family name (such as ITC Barcelona), EnumFontFamilies will return all the font names (such as ITC Barcelona Book) even if the attributes of the specific font are not directly supported by the attributes and flags in the FONTMETRIC and LOGFONT structures.

A big problem with GDI is that it doesn't have adequate graphics support for manipulating TrueType fonts in the same way that outline fonts can be manipulated in PostScript or GPI. In particular, GDI is missing matrix transforms and path support, both of which are scheduled for inclusion in the Windows 32-bit API.

As partial compensation for this deficiency, Microsoft has supplied GetGlyphOutline. This function lets you obtain the outline curve of a particular character in a font. The outline can be either hinted or unhinted. The character can be returned as either a bitmap or polygon, and you can also specify a rotation matrix. It's a bit of work, and not as clean as I'd like, but it's there if you need it.

Finally, another new text function works with both TrueType and older fonts. This is GetTextExtentEx, an extended form of the GetTextExtent function. GetTextExtentEx accepts a **string** and a width, and reports the number of characters in the **string** that will fit in the width.

An Easier DDE?

For several years, DDE has been one of the, well, more challenging areas of Windows programming. DDE uses the normal Windows messaging system to implement interprocess communication. While no one denies the power of DDE in allowing Windows applications to exchange data, working with the DDE messages (and the various rules for memory and atom management) can be very difficult.

Windows 3.1 provides a partial solution to these problems with DDEML-the Dynamic Data Exchange Management Library. This is a function-based API layer built on top of the old message-based DDE. The message-based DDE is still available, of course, and programs that use it can carry on conversations with DDEML-based applications. However, DDEML is available only in protected mode.

Although DDEML ultimately promises to make DDE coding easier and safer, approaching DDEML after learning the intricacies of message-based DDE can be daunting. DDEML consists of 26 new function calls, 9 new structures, and 142 new defined constants, 16 of which are transaction types that are received by a callback function.

Much of the DDEML terminology is the same as message-based DDE. ADDE conversation still occurs between a server (the program that has data) and a client (the program that wants this data). Other terminology is slightly different. For example, conversations in message-based DDE are based on an application name, topic name, and item name. With DDEML, this has been changed to a service, topic, and item name.

Following is a brief overview of some of the new DDEML functions and protocols.

Any program that uses DDEML (as either a client or a server) calls DdeInitialize early on to register a callback function and specify possible transaction filters. DdeInitialize returns an instance identifier used with all other DDEML functions. When a program is finished using DDEML, it calls DdeUninitialize.

The DDE callback function has eight parameters, the first of which is a transaction type. Like a window procedure, the DDE callback function uses a large switch statement for processing transactions. Most of the other parameters to the callback function are transaction-specific.

As with message-based DDE, the service, topic, and item names are text **strings** . However, these text **strings** are not passed directly to DDEML functions. Rather than using global atoms to represent **strings** , a DDEML program uses five new functions that work with **string** handles. The DdeCreateStringHandle function accepts a **string** and returns a **string** handle, and DdeQueryString accepts a **string** handle and returns, first, the length of the **string** and, second, the **string** itself.

As with global atoms, all DDEML **strings** are stored in a global **pool** . (In fact, DDEML uses global atoms internally for these **strings** .) **Strings** are not repeated in this **pool** and each **string** is associated with a usage count. But **string** handles are unique-the same **string** can be referenced by two different **string** handles. To determine if two **string** handles reference the same **string** , the program uses DdeCmpStringHandles.

When DDEML passes a **string** handle to a DDE callback function, it

becomes invalid when the callback function returns. To keep the **string** handle around a program can use DdeKeepStringHandle. The usage rules for **string** handles are somewhat easier and more consistent than when working with global atoms in message-based DDE. DdeCreateStringHandle and DdeKeepStringHandle increment the usage count, and DdeFreeStringHandle decrements the usage count. When the usage count drops to zero, the **string** is removed from the table. The increment and decrement calls should of course be balanced in each DDEML application.

DDEML also implements six new data-management functions on top of the standard Windows global memory functions. DdeCreateDataHandle accepts a pointer to data, allocates a global memory block, copies the data in, and returns a data handle for the block. When this handle is passed to a callback function, a program can copy the data into local memory using DdeGetData. The receiving program can also access the data (without copying it) using paired calls to DdeAccessData and DdeUnaccessData. When the program that created the data block is finished with it, it calls DdeFreeDataHandle. The only other data management function is DdeAddData, which can add data to a data block after it has been created but before it has been passed to a DDEML function.

The DDEML **string** and **memory management** functions and rules are simpler and more rational than the corresponding code required for message-based DDE.

A client application initiates a DDE conversation by calling DdeConnect with its instance handle and **string** handles for the desired service and topic names (either or both of which can be NULL). The server callback function then receives a transaction type of XTYP_CONNECT or XTYP_WILDCONNECT if one or both **string** handles are NULL. The server can return TRUE if it supports the topic or FALSE otherwise. If it returns TRUE, then the client's DdeConnect call returns a conversation handle. Later on, the DdeDisconnect call terminates the conversation.

Next, the client application generally uses DdeClientTransaction to request data from the server, specifying a transaction type for a one-time request (XTYP_REQUEST)oranadviselink(XTYP_ADVSTART). The server's callback function receives this transaction. For an advise request, the server calls DdePostAdvise to notify the client when the data has changed. The client's callback function then receives a XTYP ADVDATA transaction.

While some of DDEML may appear as simple function calls that hide the DDE messaging, it's more than that-some of the problems with DDE involve time-outs when a request is not answered immediately. DDEML also hides this time-out logic. DDEML also implements support for multiple conversations, and the system registering of supported service names by a server to reduce message traffic.

Object Linking and Embedding

If you thought DDE was fun, wait until you get a load of Object Linking and Embedding (OLE, pronounced either as O-L-E, or "oh-lay"). OLE involves 64 new function calls, and is undoubtedly the largest and most complex single enhancement to Windows 3.0. The OLE API certainly deserves an article of its own, which will get written as soon as somebody deciphers the documentation and figures out how to use it.

OLE shifts the user's focus from applications to documents by defining a format for compound documents. These can contain multiple forms of data that are understood and managed by multiple applications. This allows applications to concentrate on tasks they do best, and end users to use various combinations of these applications to construct a particular document. OLE is best understood by comparing it with the other two methods through which Windows applications can share data.

When using the clipboard, a program obtains data from another program in a standard format, generally ASCII text, a bitmap, or a metafile. The data exists only as datathere is no link to the program that originally put the data in the clipboard.

When using DDE, a program also obtains data from another program, and that data is also in a standard format (text, bitmap, or metafile). However, the client program can maintain a link to the server program that delivered the data. If you change the original da in the server program, it can also be updated in the client program.

OLE also enables one program to obtain data from another. However, this data can be in two formats: one format is understood only by the program sending the data; the other is a display format (usually text, bitmap, or metafile) for the receiving program to represent the data on the

screen.

The classic example is a small spreadsheet that is brought into a word processing document. When transferring the spreadsheet through the clipboard, the spreadsheet becomes simply a block of static text that has lost all connection to the spreadsheet program. With DDE, the word processing program has the spreadsheet in a text format, but it maintains a link to the spreadsheet program. If you start up the spreadsheet program and change the spreadsheet, it will change in the word processing document.

With OLE, the word processing program holds the spreadsheet in two formats-the original spreadsheet data format known only to the spreadsheet program, and a text version for display. When you select the spreadsheet in the word processing program, the spreadsheet program is invoked, the word processing program hands it the spreadsheet data, and you can use the spreadsheet program to change the spreadsheet. When you end the spreadsheet program, you're back in the word processing program with the updated spreadsheet.

The power of OLE is better illustrated by a different example. Suppose you want to attach a voice annotation to a word processing document. You have a program to record and play back sound (let's call it RECORDER) but your word processor knows nothing about the format of the sound. You can record a voice annotation with RECORDER and transfer it to the word processor. The word processor stores the data in the document in two formats-the digitized sound (which it knows nothing about) and a display format (perhaps an icon showing a pair of lips). The word processor displays the icon. When you select the icon in the word processor, the RECORDER program is invoked (perhaps it need not even be visible), the word processor passes it the digitized sound data, and the RECORDER program plays it.

Thus, with OLE, applications can become specialized tools. A single document can contain a variety of data in a number of unique formats, where each format is associated with a particular application. As you can probably surmise, OLE is a complex system with lots of new concepts.

Hassle-Free Dialogs

For users, one of the most familiar parts of a Windows application is the File Open dialog box. Users know that typing Alt-F-O always displays a dialog box that lists files to load into the application.

For programmers, writing a File Open dialog box is a holy terror. It requires a lot of **string** parsing and validation not necessary in most other dialog boxes. Many programmers have concluded that the File Open dialog box is so common it should be built into Windows itself.

Now it is. The new COMMDLG.DLL library exports nine functions that may save you unbelievable amounts of time. Not only is there a GetOpenFileName function that invokes a File Open dialog box (see Figure 9), but there's now a GetSaveFileName for a File Save dialog box. Another common dialog box is for printing, and COMMDLG also includes a PrintDlg.

To allow users to customize colors and fonts in an application, COMMDLG has two more standard dialog boxes-ChooseColor and ChooseFont. ChooseColor lets the user choose and customize colors using either the RGB (red-green-blue) or HLS (hue-lightness-saturation) color models. The ChooseFont dialog box can differentiate between TrueType and non-TrueType fonts and (if desired) limit the selection to TrueType only (see Figure 10). For word processing and text editor programs, COMMDLG also includes FindText and ReplaceText dialogs.

On first encounter, you may find these functions quite complex. This is because there are many options for customizing the dialogs for your particular application. However, using them is a lot easier than writing these dialog procedures yourself Printing Without Escape

The Windows Escape function always seemed a bit odd. Sure there are occasions when you might want to bypass GDI and go directly to the printer driver to take advantage of printer features not directly supported by GDI, but Escape is also required for normal printing-to start and end printer documents and delineate pages.

One problem with Escape is that the parameters for passing data between an application and printer driver are declared as far **string** pointers. Anything else has to be cast to a far **string** pointer, nullifying C type-checking.

Windows 3.1 includes six functions to print without Escape, plus an additional new printer function. The new StartDoc, SetAbortProc, AbortDoc, and EndDoc functions supersede the corresponding Escape functions. You

should now use the StartPage and EndPage functions rather than the NEWFRAME an BANDINFO Escape calls. The superseded Escape functions are still supported for backward compatibility, of course.

Another new GDI function is ResetDC. You can call ResetDC only between printer pages (that is, not between a pair of StartPage and EndPage calls) to change the printer device mode. This typically will be used to change page orientation between portrait and landscape in the middle of a print job, something that has been impossible.

Over the years, Escape has become a virtual catalog of everything that GDI can't do. Fortunately, an enhanced GDI in the Windows 32-bit API promises to eliminate many of these functions. Windows 3.1 makes a good beginning by eliminating the Escape function for routine printing.

Stress and Death

If you were trying to build a perfect programmer, you'd probably want to subject this strange being to stress and see how he or she reacted. For a programmer, stress can be caused by a flickering monitor, C compilers that begin reporting "out of heap space" errors at 3:00 AM, pizza vendors who forget the extra cheese, or the OLE documentation.

If you were trying to build a perfect application program, you would also subject it to stress-low (or no) memory, no disk space, or no file handles. The new STRESS.DLL library contains functions that artificially create these conditions so you can see how your program responds. Because this facility is for testing, STRESS.DLL will be distributed with the Windows 3.1 SDK rather than the retail Windows release. STRESS.DLL exports ten functions.

AllocDiskSpace creates a large file, leaving an amount of disk space specified in the function call. AllocFileHandles allocates file handles, again leaving a number specified in the function.

STRESS.DLL contains three functions for testing low memory conditions. AllocMem allocates global memory, AllocUserMem allocates memory in the USER.EXE local heap (where resources and window information is stored), and AllocGDIMem allocates memory in the GDI.EXE local heap (where GDI objects are stored). In each of these three functions, you specify a contiguous memory size that the function should leave free. Five additional functions in STRESS.DLL return everything back to normal.

I have a funny feeling that testing program stress is likely to increase programmer stress. For example, when was the last time you checked if a Create Pencall failed? Do you even know what happens? It's easy to find out by calling AllocGDIMem before CreatePen. You'll find that nothing really bad takes place. CreatePen returns NULL, of course, and if you eventually select the null handle into the device context using SelectObject, SelectObject returns NULL to indicate an error. The device context is left with the pen object previously selected.

Another way to use the STRESS.DLL library is to write a little program that uses a Windows timer to randomly create low memory or disk space conditions. You'd run this little stress program whenever you're testing your code. It should make things fun for your beta testers. The inner Sanctum

What's really going on inside Windows? Does the thought intrigue you or frighten you?

In the past, programmers who needed to know some of the internal data inside Windows had to rely on undocumented methods for getting hold of it. The new TOOLHELP.DLL library exports 38 functions that let the programmer find out about the whole Windows environment and implement debuggers more easily.

TOOLHELP lets you obtain information about all the tasks and modules running under Windows, all the window classes currently registered, all the global memory blocks, and local heaps associated with applications and DLLs. Debugger support includes task switching, locking input, and interrupt and exception handling.

Drag and Drop

After struggling with the other new APIs in Windows 3. 1, you're due for a break, and drag-and-drop provides it. This is a very simple AIPI that consists of only four new function calls and one new message. True, the functionality is simple also. The drag-and-drop API lets the user grab a file (or group of files) from the File Manager and drag them over to your application. Your program gets the filenames.

First, a program calls DragAcceptFiles (probably during window initialization) to indicate that the window can accept files dragged from

the File Manager. If the user subsequently drags a file (or files) from the File Manager to the program's window, the program receives a WM_DROPFILE message. The wParam message parameter is a drop handle used in the other three function calls.

The program then uses this drop handle to call DragQueryFile-if necessary, multiple times-to get the filenames of the dropped files. The DragQueryPoint function obtains the mouse pointer coordinates relative to the application window. When finished, the application calls DragFinish to deallocate memory used during the drag-and-drop operation.

This is so clean and simple an API that you should definitely include it in any program that works with files. installation Aids

Windows programs are getting much more sophisticated and, consequently, larger. It's not surprising these days to see products shipped on half a dozen or more diskettes. To reduce the number of diskettes, several software vendors have begun shipping their applications in compressed file formats. During installation, these compressed files are expanded for storage on the hard disk.

The Windows 3.1 SDK makes this process a bit smoother by including DOS-based COMPRESS.EXE and EXPAND.EXE programs. The COMPRESS program shrinks files using a Lempel-Ziv compression algorithm and the EXPAND program expands them to their original form.

Vendors wishing to develop a Windows-based installation program can compress their application files for diskette storage using COMPRESS.EYE, and then take advantage of the LZEYPAND.DLL library during hard disk installation. This DLL exports ten functions that implement Lempel-Ziv expansion algorithms.

For example, to decompress a single file, an installation program can open the compressed source file and the destination file by calling LZOpenFile. Calling LZCopy copies the file and uncompresses it in the process. The installation program then calls LZClose to close both files. LZEXPAND also includes functions to copy multiple files, or to work with files that contain a combination of compressed and uncompressed data.

Windows 3.1 maintains a registration database containing information about applications installed under Windows.

This is somewhat similar to the information format in INI files, but in a more flexible multilevel hierarchy. The information assists in integrating applications via DDE and OLE, and makes it easier for the File Manager and Print Manager to get information about other programs. Seven functions are devoted to the registration API. Windows 3.1 also includes a few enhancements to the DDE command- **string** feature for installing applications in a program group in the Windows Program Manager.

When multiple programs share DLLs, problems result when the DLL is updated. A user could install an application that replaces a DLL with an older version. A new "version" API assists in identifying DLL versions and avoiding this problem.

To Point One or Not to Point One

Windows 3.0 had such a strong impact on the PC industry that there was much incentive for Windows developers to upgrade their existing code to 3.0 or for developers new to Windows to port their DOS programs to 3.0. With the introduction of 3.1, however, there will probably be a significant number of users who continue to work with Windows 3.0.

This poses a dilemma for developers. Should you go ahead and take advantage of 3.1 features now or should you wait?

Fortunately this is not an either/or question. You can take advantage of some 3.1 features and have your programs run under Windows 3.0. This is part of the magic of dynamic linking. You can make use of OLE, DDEML, the common dialogs, and version-checking features, and ship the required DLLs with your application.

The most important step is to test that your 3.0 code can take advantage of TrueType fonts. This should not require significant changes to your code (if any), although you may also want to replace your font selection dialog with the one in COMMDLG.DLL.

And remember-don't plan your vacation just yet. As soon as you assimilate Windows 3. 1, it'll be time for 32-bit Windows! Stay tuned.

CAPTIONS: Function calls in Windows and Windows extensions. (table); Encoding of the nFontType parameter to the EnumFonts callback function. (table); Encoding of the lfPitchAndFamily field of the TEXTMETRIC structure. (table)

SPECIAL FEATURES: illustration; table
DESCRIPTORS: Enhancements; Applications Programming Interface; GUI; Functions; Font Package; Program Development Techniques
SIC CODES: 7372 Prepackaged software
TRADE NAMES: Microsoft Windows 3.1 (GUI)--Design and construction
OPERATING PLATFORM: MS Windows
FILE SEGMENT: CD File 275

**4/9/2 (Item 2 from file: 275)**


01346253 SUPPLIER NUMBER: 08102586 (THIS IS THE FULL TEXT)
**FoxPro! (Software Review) (includes related article on FoxPro's memo fields, and a related article on changes to data files fromFoxBASE to FoxPro) (evaluation)**
Hawkins, John L.
Data Based Advisor, v8, n2, p70(15)
Feb, 1990
DOCUMENT TYPE: evaluation ISSN: 0740-5200 LANGUAGE: ENGLISH
RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 8318 LINE COUNT: 00652

ABSTRACT: FoxPro is an outstanding addition to the data base management system market. It features a new interface, good documentation and on-line help, an internal editor that provides sufficient program writing capability for many users, a report writer that allows creation of complex reports and labels without additional programming, and new commands and functions, including CURDIR, which returns the current DOS directory on any drive. Speed is estimated at from one to as much as 16 times greater than other data base management systems. FoxPro's biggest drawback is its memory consumption. Connectivity will be provided by FoxPro/LAN, due to be released early in 1990. FoxPro is $795 for a single-user version, $1,095 for FoxPro/LAN, and an additional $500 for an unlimited runtime version. Upgrades from FoxBASE+ to FoxPro are $195, to FoxPro/LAN from FoxBASE+/LAN $250, and from FoxBASE+ Runtime to FoxPro Runtime $50.

TEXT:

FoxPro!

Wow! I wish it would run on my PC! When Fox Software previewed their unprecedented FoxBASE+/Mac 2.0 in early 1988, the reaction of PC users was predictable. FoxBASE+/Mac merged the data manipulating, application-building power of the dBASE language with an innovative, Mac-like user interface, then embellished the package with sophisticated developer tools. "Too bad it takes a Macintosh to support such a product." Or so we PC users thought.

A few months later, Fox Software disclosed--exclusively to Data Based Advisor readers--details of an even more advanced version, FoxPro, under development for PCs running MS-DOS. Contrary to popular wisdom, Fox was convinced the event-driven Macintosh approach could be implemented on a character-based computer. In Sept. 1989, Fox revealed FoxPro to over 600 people attending the first Fox Developers Conference, and sent them all home with a pre-release version for final beta testing. On Nov. 9 Fox officially released FoxPro, then issued a Nov. 16 update to correct a production error in the original release. Because Fox has a practice of implementing program corrections whenever needed, the FoxPro you buy may well have a later date. So FoxPro is here, and here, and here. This review is based on the Nov. 29 release, though all versions are identified as FoxPro 1.0. (To see the actual release date, type "? VERSION(1)".)

In FoxPro, Fox Software has raised the dBASE standard well beyond Ashton-Tate's high water mark. Users, developers, competitors, and even those who have pronounced the dBASE standard "dead" must acknowledge the FoxPro achievement. Fox has given dBASE technology a new direction, a new life, and perhaps a future. Nor is FoxPro 1.0 the ultimate extension of dBASE technology. Fox plans exciting additions for FoxPro 1.1, FoxPro 2.0, and beyond. Elsewhere, Nantucket, the other major force pushing the dBASE platform forward, advances with a different and extremely powerful metaphor

in the coming Clipper 5.0. Ashton-Tate, much bigger than Fox and Nantucket combined, is patching up a leaky dBASE IV. (A look at pre-release dBASE IV 1.1 shows a much-improved product that may be quite acceptable in its own right.) It's much too early to declare a winner (they all might be), or even a best choice. But it's not too early to foresee FoxPro becoming a potent product for both developers and users. Unless Fox fails to meet their own specifications, FoxPro has unlimited user-pleasing potential.

If you're involved in the dBASE world, whether you feel "safe" with Ashton-Tate's dBASE, satisfied with FoxBASE+, or empowered by Clipper, you should evaluate FoxPro. If you use a product like Paradox or R:BASE because dBASE is "too hard," the power of dBASE may now be reachable through FoxPro's new interface. In this and coming issues Data Based Advisor will give you extensive information on FoxPro. The same treatment will be given to Clipper 5.0 and the improved dBASE IV 1.1 when they become available. Based on information from FoxSoftware President, David Fulton, several months of working with pre-release FoxPro, and a few weeks with FoxPro 1.0's various updates, this special report is a review of what's new, what's different, and what's exciting.

What's a FoxPro?

As we explore FoxPro, comparisons to dBASE IV must be drawn. As competitive products, functional similarities are intentional, but implementations vary in important ways. Like dBASE IV, FoxPro is several things in one. Unlike dBASE IV 1.0, where components feel bolted on, most of FoxPro 1.0 is seamlessly blended into one program. In fact, the various pieces share common internal routines. For example, the same editor is used almost everywhere, whether creating an index expression, revising a previously entered command line, editing a memo field or writing a program. The exceptions are the add-on utilities FoxDoc, FoxCode and FoxView, already provided with FoxBASE+ 2.1 and essentially unchanged (unfortunately) for FoxPro 1.0. (For more information, see John Bauman's article next month.) They're expected to be improved for FoxPro 1.1. FoxView, FoxDoc and the optional FoxGraph appear as choices on the FoxPro system menu or they can be run from DOS, but as products from other vendors their look and feel differ from FoxPro and from each other. All other facets of FoxPro are internal and available only from its system menu, by direct commands typed in, or through programming.

The most obvious change in FoxPro is the interface. Gone is--shed a tear--the dot prompt. There's a flashing cursor, but no dot! Instead, you type all commands in a command window, that shows the current command line and up to a full screen of previous commands. Across the top row of the screen you'll see a horizontal menu with several menu pads, in the style of IBM's Standard Application Architecture/Common User Access (SAA/CUA) standard. Selecting a menu pad reveals a pulldown menu of choices. Some of these call additional menus, or Mac-like dialog boxes, or other interactive routines. Overall, FoxPro's menus are easy to use, although some combinations of operations require a bit of jumping around. Most FoxPro functions can also be invoked by typing an equivalent dBASE language command. As much as possible FoxPro eliminates the need to type. Just select from menus. And if you can find space on your desk for a mouse to wriggle, FoxPro even eliminates a big chunk of keyboard use. Like every other mouse-oriented program I've tried, I find FoxPro easier to use with a mouse--like it or not. This isn't really a fault of Fox or others trying to combine mouse and keyboard interfaces. Because a mouse user can dart to various locations in a dialog box and around the screen, FoxPro can't be locked into a rigid linear progression of menu choices. The flexibility to jump across the screen and select from several unrelated options speeds up program use tremendously. Providing ways to handle such "eventdriven" actions from the keyboard requires the use of more key combinations. The assigned keystrokes are not always intuitive, and are different from the FoxBASE/dBASE/WordStar "standard" (a keyboard macro facility allows optional use of many FoxBASE-compatible key assignments). Some of the eventdriven mechanism used in the FoxPro interface is also available within applications, with more to come.

Though FoxPro resembles dBASE IV in the use of SAA/CUA-style menus, beneath that first layer FoxPro is definitely different. FoxPro uses "true" windows for almost all screen elements. These windows are treated like objects, meaning they can be manipulated independently of other screen elements. Most interactive pieces of FoxPro appear on screen in windows, which can be moved, sized, zoomed to full screen, or hidden from view.

Windows can overlap and even hide behind each other. Similar windows are available to developers. Because of the window approach, the menu system, and enhanced interactive editing tools, non-programmers can do quite a bit without knowing a single FoxPro/dBASE command. For those who do know some of the language, it's quite easy to intermix menu choices and direct commands. In fact, selecting a menu choice causes the corresponding language command to appear in the command window, which is a great way to learn FoxPro. It's even possible to run through a series of menu operations, allow the equivalent commands to accumulate, then cut and paste the command sequence into an editing window for inclusion in a program. A few activities available through the system menu, such as the calculator and calendar/diary, may not be especially usable in programs until Fox adds additional control and system variable access (not promised but obviously needed).

FoxPro can read DBF data files created with FoxBASE+, Clipper, dBASE III PLUS, and dBASE IV. But the reverse is not true; other products cannot understand FoxPro's enhanced DBFs. A DBF that does not use either the FoxPro float type numeric field nor FPT-format memo field should be readable by all of these products. A future update of FoxBASE+ may be able to recognize FoxPro-specific DBFs. FoxPro uses FoxBASE+-compatible IDX indexes and will automatically copy dBASE indexes to FoxPro format (the original dBASE indexes are thereafter abandoned). Memo fields are supported in both the common DBT format and the new, improved FoxPro FPT format (see my memo field article). FoxPro reads standard PRG and FMT source code files and compiles them into its own runtime FXP and PRX formats. It can't use compiled FOX FoxBASE+ or DBO dBASE IV program files. FoxPro can use FoxBASE+ and dBASE III PLUS FRM report definition files, and optionally convert them to FRX format.

Since setting up a session involving several database files, indexes, relations, and browse windows is a chore, FoxPro provides several mechanisms for saving and restoring such environments. CREATE/SET VIEW saves and restores the current status of all open DBF and NDX files, including RELATIONS, using a VUE file. Some activities, like BROWSE, are saved in FoxPro's hidden Resource file, FOXUSER.DBF. BROWSE LAST restores the browse window as it was last exited. BROWSE PREFERENCE saves and restores a browse session according to a predetermined combination of parameters. Environment settings needed to run a specific label can be saved and restored using an LBV file.

Against the clock

As always, Fox considers speed to be a fundamental requirement. They estimate FoxPro to be 2.6 times faster than FoxBASE+ 2.1, about two times faster than FoxBASE+/386, eight times faster than dBASE IV, and almost 16 times faster than dBASE III PLUS. Some FoxPro facilities are not up to its generally lightning-fast performance. For example, the dBASE IV-style menu commands are powerful, but the results can feel relatively slow, probably due to **memory management** overhead. FoxPro is noticeably sensitive to the number of memory-consuming features activated (like windows, menus, and browse sessions), and to the amount and types of RAM available. As such products get more complex, determining what to compare and how to measure becomes more difficult, so it is impossible to declare FoxPro "the fastest." But in key areas I'd definitely call it jet-propelled. (We'll be featuring our own performance tests in an upcoming issue.) For top speed: avoid FLUSH, SET DOHISTORY OFF, don't go crazy with windows, and free up memory.

Memory loss

FoxPro seems to require about 480K-plus to do anything useful, and much more to do anything really sophisticated. In fact, it isn't too hard to create a program that simply won't run in 560K. For maximum performance, give it as much RAM as possible. Standard RAM is most important, so hardware add-ons like the Maxit board (from Osborne McGraw-Hill Software and recommended by Fox Software) and certain software utilities that make unused reserved area RAM available can be worthwhile investments. With some computer and video combinations it's possible to extend a 640K machine to 704K or more. FoxPro will use as much expanded memory (LIM/EMS) as it finds, and is claimed to double in speed with enough EMS. While FoxPro should run on an XT-class PC, a complex application with windows and pop-up menus can appear jerky. A 386 computer is an ideal environment, since EMS memory can be added inexpensively by using software drivers like QEMM- 386 (from Quarterdeck Systems) or 386Max (from Qualitas). These drivers make

unused reserved memory available, and convert extended to expanded memory without special hardware. FoxPro directly uses all 386Max memory and includes a utility to take full advantage of the extra memory QEMM makes available. (Fox says QEMM is three times faster at EMS access than 386Max.) FoxPro is also compatible with QEMM's partner, DESQview (sold together as DESQview 386), and it will take advantage of a math coprocessor.

If you're familiar with such stand-alone Fox programs as FoxDoc, FoxView, and FoxGraph, you may wonder how they can be run from the FoxPro system menu. Where does FoxPro "go" while these large external programs load and run? It gets "FoxSwapped." When any external program is to be run, whether a Fox utility or your own using the RUN command, FoxPro uses its FoxSwap memory manager utility to free up enough RAM. Just specify the amount of memory needed to execute the external program, and FoxPro will temporarily remove enough of itself from memory. To make as much memory as possible available, specify "RUN 0 <program name>". FoxSwap is even usable with FoxBASE+ to run a large external text editor.

The written word

FoxBASE+ is known for sparse, inadequate manuals. The much-improved FoxPro documentation includes several books: Users Guide, Commands & Functions, FoxView/FoxCode/FoxDoc, and Quick Reference. A comprehensive online help system is available within FoxPro. And instead of FoxBASE+'s recommendation that "any textbook for dBASE III PLUS can also be used as a reference for FoxBASE+," FoxPro comes with its own printed tutorial, demo program, and practice files! Learning FoxPro isn't so much difficult as it is different from predecessors like FoxBASE+ and dBASE III PLUS. Those familiar with dBASE IV will feel most at home, though FoxPro may take anyone some time to absorb. It is time well spent. Technical support, struggling to keep up with Fox's rapid growth, is available free with a call to Toledo, Ohio. Fox tries to direct tech support calls to a FoxPro group, but getting a return call has been spotty. Fox also maintains a forum on CompuServe (PCVENA). FoxPro is branded with a serial number but isn't copy protected.

For the FoxProfessional

For developers, programmes and other power users, FoxPro creates a dilemma. First, while the interface and several key functions are greatly improved, there's a lot to learn about all that is new. Second, as database application languages become more complex, it becomes more difficult to be fluent in more than one dialect. FoxPro has more than 500 commands, functions, and system control variables, with hundreds of additional optional clauses. To fully absorb FoxPro, you may find yourself unable to spend the time to remain proficient with other products like dBASE or Clipper. Is it worth it? Should you choose FoxPro over dBASE?

dBASE IV 1.0 isn't even a contender. Independent developers familiar with both products are already proclaiming FoxPro's features and implementation to be superior to the approach taken by dBASE IV 1.1. (Some people who know the current state of un-released dBASE IV 1.1 estimate that FoxPro represents at least a one-year lead in programming achievement.)

As promising as FoxPro is, it is evolutionary, not revolutionary. Becoming a FoxPro requires great faith in the future of Fox Software. To attain and maintain leadership Fox has to prove it can consistently deliver the goods. Some beta testers were disturbed when Fox appeared to follow in Ashton-Tate's footsteps, releasing FoxPro with unresolved bugs. Other testers had no problems and said FoxPro was rock-solid at release. My staff knew of a few bugs that remained unresolved as of early December. Perhaps the contrasting opinions among beta testers resulted from the differing combination of FoxPro capabilities we used. Though both products were released with anomalies, the difference between FoxPro 1.0 and dBASE IV 1.0 might be described as stability and strategy. The types of bugs found in early release of FoxPro seem to be consistent, meaning they can be identified and worked-around (and, we hope, corrected). dBASE IV's quirks appear randomly, and cause totally unpredictable crashes. According to dBASE IV trainer Adam Green, you can't program around the problems in dBASE IV 1.0; Green finds no similar instability in FoxPro 1.0 (and has just added FoxPro to his classes). FoxPro bugs seem to be minor, and Fox's solution is to try and fix them within days of discovery, providing a free update to anyone who reports a problem. When enough bugs have been excised, Fox plans to send a mass update to all users. Early FoxPro users are remarkably tolerant of this approach, explaining that all 1.0 products of this complexity are imperfect. What counts, they say, is that FoxPro is

fundamentally sound, and any problems will be resolved quickly. Ashton-Tate has not pursued a similar fix-bugs-as-we-find-them tactic, and has been unable to provide users with a repaired dBASE IV in over one year.

Fox has ambitious plans, and Nantucket is certainly in its sights too. But FoxPro isn't a total replacement for Clipper--yet. Clipper is extremely powerful, and release 5.0 will eliminate some major limitations (though Nantucket didn't meet its announced release date). Other elements of Nantucket Future Technology (NFT) promise to extend the power of Clipper's advancement over dBASE programming in ways FoxPro may never approach. On the other hand, even hard-core Clipper-heads (as some like to call themselves) may be tempted to switch projects to the extremely productive interactive development environment FoxPro offers. Software developers, like slalom skiers, are always racing the clock. It's a rare client who will say, "Take your time. I'll pay whatever it costs." Mostly, programming projects are constrained by fixed budgets and critical deadlines. So anything that gets the job done faster and better is money in the pocket. FoxPro offers an efficient application development environment unprecedented in the dBASE world.

There are many ways to set up FoxPro to aid coding and debugging. I'll describe my typical set-up as an example. Using a VGA monitor system, I enter FoxPro and issue the command SET DISPLAY TO VGA50 (or I do it automatically via the CONFIG.FP file). Since a standard PC screen is 25 lines, when I run my application, it fills just the top half of the screen. In the bottom half I open trace, debug, editing, and command windows, all sized and moved to fit. As the program executes, I can watch it unobstructed while also watching the program code display line by line in the trace window. The live, changing values of any variables and expressions that need analysis are displayed in the debug window. I can slow down or single step the program as needed and set break points in the code. When a break point is reached, the program crashes, or I suspend with Esc, I can pop directly into the editor window, make changes to my source code, then rerun the program (from the top, since there is no way to edit the active PRG and resume execution). FoxPro will automatically recompile the changes to disk. Or I can move to the FoxPro system menu and take action, or issue commands in the command window. All of this occurs on a single screen with everything displayed simultaneously. Characters on a VGA 50-line screen are a little crowded, but I can do so much so fast that a project really moves along. Even without a monitor system that can display 50 lines (or 43 lines with EGA), FoxPro's multiple windows save time. Just shrink them down a bit and overlap less- important areas of the application screen. Or tell FoxPro to invoke the trace window only if an error or breakpoint is encountered. In theory, you'll never have to leave FoxPro to write and debug a complete application. (Though FoxPro's slow, awkward editor might make it hard to stay in FoxPro when working on a sophisticated application.)

Using Clipper--even Clipper 5.0 with its promised large improvements in linking and debugging--this unified approach is impossible. (Nantucket president Larry Heimendinger has told me Clipper won't be providing any form of interactive development environment.) To develop quickly in Clipper and FoxBASE+ I resort to two networked computers, one to run the application, the other to edit and compile it. For me, FoxPro liberates one entire computer.

FoxPro also liberates many previous limits, although Clipper remains the champion of the boundless. FoxPro supports 25 open database files with 25 total open indexes. Up to 99 files of all types can be open at once (subject to DOS). A record can have 255 fields, with 254 maximum characters per field, to 4,000 total characters per record Numeric fields can store up to 20 digits, including decimal and sign, with 16 digits of numeric computational precision. Memory variables can range up to 3,600. DO calls can be nested 32 levels, READs can be nested four levels. A FoxPro program command line is limited to 1,024 characters. Maximum individual compiled program module size is 64K per FXP file (meaning a PRG of around 100K). Up to 64 structured programming commands can be active at one time. Up to 25 BROWSE windows can be open at once, and the total number of open windows is limited only by memory. Memory is definitely a limitation. Experimentation reveals that a real application runs out of free memory long before most of these limits are reached. For example, each BROWSE session takes 16K. A network application that opens 10 simultaneous BROWSE windows may be maxed-out (if you have a reason to do so). Each file loaded in the text

editor requires two file handles, meaning the DOS FILES=statement in CONFIG.SYS may need to be increased, further reducing memory available to FoxPro. It is clear that sophisticated programming with FoxPro requires a more structured approach to **memory management** . Even though dBASE III PLUS and FoxBASE+ programs should run just fine, programs written specifically for FoxPro should adopt a completely different structural approach. Clipper programmers may already be familiar with the modular, procedural style FoxPro allows, an approach that actually makes application development and maintenance easier. (Here are some hints: put common code in subroutines in the form of procedures and functions, and don't leave things in memory you don't immediately need.)

The editor

When I say "built-in editor," do your eyes glaze over? If you're thinking of those clumsy beasts contained in dBASE III PLUS and FoxBASE+, think again. FoxPro's internal editor, while not a threat to the popularity of professional editors Brief/dBrief and QEdit, has much of the program writing capability many people will need. (In fact, I wrote a large portion of a book using FoxPro's editor and memo fields!) The same editor engine is invoked any time you need to edit anything. Called by pull-down menu, by typing MODIFY COMMAND or MODIFY FILE, or by the debugger during testing, FoxPro's editor provides find/replace (though primitive), cut/paste, variable tabs, auto-indentation, and full undo/redo for the entire session. You can cut and paste almost anything between open files; FoxPro even includes a screen capture cut/paste facility. Specific editing defaults can be defined for different file extensions. There's no limit on line and file length except available disk space. Like most FoxPro objects, the editor appears in a window. This means it can be sized and moved freely (as I do for my 50-line screen set-up). In fact, you can open multiple files in multiple edit windows, making it easy to work on an entire application. You can open files by wildcard, and even type MODIFY COMMAND *.PRG (BRIEF can do this too). A RANGE option will even move the cursor at specified characters within the edit window. The editor can tolerate null and end-of-file codes, making it possible to edit any type of file. It's even possible to load in a DBF, edit a field name, then save the DBF back to disk, changed but otherwise unharmed (this isn't the recommended way to change field names!). Fox plans to continue to improve editing facilities as FoxPro evolves. (The editor is missing some obvious functions, like cursor line/column information and smart source code formatting. The file opening dialog box annoyingly defaults to "database", not the "program" files developers will be using. And the editor is noticeably slower than Brief and QEdit.) By the way, if you don't like FoxPro's editor, just tell it to automatically use your own.

Refining the language

I find it more and more difficult to talk about the "dBASE language." The dBASE III PLUS of 1986 may be the last product to universally define the core language. Since then each compatible product has extended the language differently. FoxPro 1.0 claims to be 100-percent language compatible with FoxBASE+ 2.1 and dBASE III PLUS, mostly compatible with dBASE IV 1.0, and somewhat compatible with Clipper Summer '87. Where dialects differ, programmers can often just change a keyword or rewrite a command line to convert a program. But a FoxPro application that uses all of its power won't be readily translatable into any other dialect. For example, FoxPro supports dBASE IV's window and menu commands, but not the other way around. Because Clipper is so extensible, much of what a FoxPro application might do could be implemented in Clipper through UDFs and third-party libraries--with enough time and money. But no single dBASE language product offers the breadth, depth and sophistication of FoxPro's native functionality. Of the 500-plus commands and functions (and hundreds of optional clauses), the FoxPro language adds more than 200 not found in FoxBASE+, over 150 not found in dBASE IV, and over 100 not found in either. These enhancements take many forms. Several bring FoxPro to near-parity with Clipper and dBASE IV, while many aspects are notably improved: windows, menus, browse, memo fields, mouse control, reports and labels, and user-defined functions. Only a few fall short. Some FoxPro features included simply because dBASE IV offers them are unnecessary or even useless.

Even though FoxPro and FoxBASE+ act like interpreters, both require that source code PRG files be compiled before execution. (If necessary, FoxBASE+ will do this invisibly and temporarily to RAM; but it does

happen.) If an up-to-date compiled FXP file doesn't exist on disk, FoxPro will automatically compile one from a PRG, visibly, intelligently, and permanently. While a third party "make" utility is desirable with FoxBASE+, FoxPro has one built-in. When a program file is called, FoxPro compares PRG date and time to the compiled FXP version (if any). FoxPro automatically recompiles the PRG if appropriate, then executes the FXP. Automatic compilation can be deactivated (SET DEVELOPMENT OFF), and compiling can also be done manually. Errors generated during a compile can automatically be sent to a corresponding ERR file for review. Compiling is only available from within FoxPro, so consultants who make PRG changes in the field for runtime environments will have to adopt a different approach than carrying the FoxBASE+ FOXPCOMP program around (maybe Fox will provide a standalone FoxPro compiler utility?). FoxPro FXP compiled files are larger than FoxBASE FOX files, mainly due to the inclusion of additional debugging code. Fox has a procedure to remove this extra code from the final version of an application.

In addition to 25 open databases, FoxPro uses two "hidden" DBF databases that developers can take advantage of. The on-line help system, called with F1, is a database that can be revised, or even substituted to add context-sensitive help to an application. A "resource file" database stores FoxPro defaults as well as programmer-specified parameters for browse windows, colors, and more. As with FoxBASE+, FoxPro allows multiple relations, but with more flexibility. To disconnect some but not all of a complex relationship, use SET RELATION OFF INTO... A new function, RELATION(), reveals the active relationship expression. RECNO(0) can be used as a softseek function to determine the next highest record when a SEEK fails.

Whatever your video display system, FoxPro can probably support its tricks, including standard 25x80, plus EGA43 and VGA50-line modes, in any imaginable color combination. SYS(2008) determines the type of video adapter and monitor in use, and SET DISPLAY TO sets the video mode. FoxPro's color control methods respond to the SET COLOR TO command. But with so many new display components, it's easier to define all elements once, then use the SET COLOR OF SCHEME command to make global changes that would require 10 of the older command lines. FoxPro also includes a comprehensive interactive color selection screen. However, the color picker isn't available from within programs.

Like Clipper, FoxPro procedures and user-defined functions (UDFs) can be embedded within any PRG (up to 1,170 per PRG), allowing modular and self-contained subroutine-style programming. FoxPro will first look for something identified as a PROCEDURE or FUNCTION in the current PRG, then in an activated procedure file, in the chain of calling PRGs, and finally on disk for a file of the same name. FoxPro can't look within PRGs that are currently inactive. Unlike Clipper, once a FoxPro PRG terminates, the procedures and UDFs it contains are no longer available. (In Clipper, any linked procedures and UDFs are globally available as long as their code segment isn't in a non-loaded overlay.) To make certain FoxPro subroutines globally available, put them in the highest level (master) PRG and/or put up to 1,170 of them in a single file and use SET PROCEDURE TO <file>. The UDF capabilities of FoxBASE+ are very limited, as are the UDFs of dBASE IV (to be expanded in version 1.1). In contrast, FoxPro allows UDFs that are almost as flexible as Clipper's, since any command can be issued in a UDF (Clipper allows UDFs to be used in more places). FoxPro allows a variable number of parameters to be passed, making it possible to create general purpose subroutines that can be called with only as many parameters as are needed. FoxPro adopts and extends the menu building tools found in dBASE IV. Complex horizontal bar, pull-down, and pop-up menus can be created and controlled through concise commands. All such menus automatically support a mouse. FoxPro's menu commands internally create and manage complicated control loops, removing them from the programmer's burden. Amazingly, I wrote a complex menu-driven multi-level application without a single DO WHILE loop! however, some developers dislike the dBASE IV menu generation approach and prefer the FoxBASE+ menu style, which is also available in FoxPro. I'm not totally satisfied with either method. Maybe FoxPro version 1.1, which will provide FoxBASE+/Mac style menus, will make everyone happy.

FoxPro provides for complex VALID clauses (which can contain UDFs), nested READs, complex INKEY() processing, and extended multiple ON KEY trapping. It supports "!" for .NOT., "!=" for <> and #, "==" for an exact comparison, and "{12/25/89}" to specify a date data type. Functions that

return an unneeded value can be preceeded by "=" without using a scratch variable. FoxPro provides two-dimensional arrays of up to 3,600 elements. With FoxPro's Clipper-like low-level file I/O functions (FWRITE(), FREAD(), and FSEEK(), you can manipulate any kind of DOS file if you know its structure. This lets programmers share data stored in foreign formats.

Making news

The FoxPro report writer allows creation of complex banded reports and sophisticated labels without programming. Headings, data, calculated fields, subtotals, groupings, headers, footers, title and summary pages, text, and even boxes can be laid out very flexibly. UDFs can be embedded almost anywhere in reports. Layouts can be previewed on screen. The system print control memory variables introduced in dBASE IV are available for further control. Unfortunately, an intelligent printer driver system similar to dBASE IV's will not appear until FoxPro 1.1. Also missing from FoxPro's report writer is dBASE IV's ability to generate actual source code. It's not that FoxPro can't produce similar reports; it just doesn't provide such reports in source code form. A major benefit to this is that users will have access to the report writer, even in runtime versions of FoxPro. On the other hand, many programmers mention report code generation as the major dBASE IV feature missing from FoxPro.

Good looking

The basic design of FoxPro's user interface begins with the dBASE IV windows concept, but FoxPro gives the programmer much more to work with. For one thing, FoxPro uses true windows where dBASE IV uses pictures of windows. This means FoxPro information can be written to the screen underneath a window, while in dBASE IV the window will be overwritten. FoxPro can even write to a hidden window object. FoxPro adds several ways to control its window objects. DEFINE WINDOW includes options to allow the user to size, zoom, close, and move a window interactively while running the application. ACTIVATE WINDOW...NO SHOW sends output to a hidden window, which can later be displayed as needed. Use HIDE WINDOW to make one or all disappear without being closed. CLOSE MEMO closes some or all memo editing windows. WLCOL()/WLROW() determine the upper left corner of a window on the screen, WCOLS()/WROWS() return the size of the window, and WONTOP() determines if it's currently on top of all other windows. WEXIST() says whether a window exists, WVISIBLE() says whether it's visible, and WOUTPUT() determines if output is directed to it. SCOLS()/SROWS() return the size of the physical screen.

Commanding presence

The following are some of the exciting new commands and functions in FoxPro:

* In addition to SCAN...ENDSCAN introduced in dBASE IV, FoxPro offers FOR...ENDFOR (with slightly different syntax than Clipper) as a cleaner way to do certain kinds of loops.

* To meet more real-world needs, TOTAL ON now takes an expression as well as just a field name.

* Very useful in accounting, GOMONTH(n) figures out the date that's exactly n months before or after a given date.

* If you've ever struggled to write a routine to let users build their own FILTER, FOR, or INDEX expressions, GETEXPR is a dream come true. It invokes the FoxPro expression builder, a much more sophisticated tool than you'd ever write on your own.

* FILTER() returns an active filter expression.

* FILER calls a FoxPro routine that resembles directory tree utilities, allowing a user to visually manage files and directories from within an application.

* CURDIR() returns the current DOS directory on any drive.

* FULLPATH() provides the full DOS path name for a file.

* PUTFILE() invokes FoxPro's Save File dialog box so the user can specify a file name.

* SCATTER...MEMVAR...EMPTY can save many STORE...TO lines of code by creating memory variables with the same name as the open database's fields. Either the record's data is put in the memvars, or they're created empty. Since this creates memvars with the same name as fields, use of the M prefix is mandatory (and generally a good idea). In addition to the dBASE standard "M->memvar" syntax, FoxPro allows the more concise alternative "M.memvar". Put memory variables back into fields with a single GATHER...FIELDS line, rather then with many REPLACE commands.

* AT() can return the starting position of the n occurrence of a

character string. **ATC** () does the same but is case-insensitive.

* ATLINE() also has a case-insensitive version, ATCLINE(), to find the line number of a string. **This** is very useful in memo fields.

* RAT() and RATLINE() determine the start of a string **in** reverse, beginning at the end.

* BETWEEN() determines if an expression falls between two other expressions, whether character, numeric or date.

* OCCURS() determines the number of occurrences of one string **in** another.

* INLIST() is a similar, redundant function.

* CHRTRAN() translates characters of a string **using** a translate table.

* STRTRAN() searches for a string **and** replaces it, just like a word processor search and replace routine.

* MIN() and MAX() work on any kind of data.

* SECONDS() returns the system time to an accurate 1,000th of a second.

* LASTKEY() returns the decimal ASCII value of the last key pressed.

* CHRSAW() determines if a character is in the keyboard buffer without affecting the buffer.

* Check and control the keyboard with CAPSLOCK(), INSMODE(), and NUMLOCK().

* PADL(), PADC(), and PADR() pads or truncates left, center, or right of a character string **to** the specified length.

* PROPER() capitalizes the first letter of each word, but isn't smart enough to be of much actual use.

* LIKE() gives the DOS-like ability to compare the contents of two variables, one with wildcard symbols * and ?.

* Two functions provide tools to identify similar sounding but differently spelled words. DIFFERENCE() compares two character strings **and** returns a number indicating their phonetic similarity or difference. SOUNDEX() returns a code representing a strings " **sound** ".

* The direct DBF editing commands BROWSE, EDIT and CHANGE have such an incredible number of options that they should find a useful new place in developed applications. Creating a BROWSE-like table view of data has always been a cumbersome programming task, but necessary since the actual BROWSE command in FoxBASE+ and dBASE III PLUS is too uncontrollable to include in an application. FoxPro's BROWSE is a new ball game, an amazing implementation, programmable in almost every aspect. This list of optional clauses should whet your appetite: BROWSE...FIELDS, FORMAT, FREEZE <field>, KEY <expr>, LAST, NOAPPEND, NOCLEAR, NODELETE, NOEDIT, NORMAL, NOWAIT, PREFERENCE, SAVE, TIMEOUT, TITLE, WIDTH, WINDOW, and COLOR. BROWSE can display calculated fields and read-only fields; field data can be validated and/or limited to a range; field headings, conditional field editing, width and data formatting pictures can be specified on a field-by-field basis; and UDFs can be embedded in a BROWSE clause. The same options can also be used with CHANGE/EDIT.

Getting along with others

Fox has attempted to make FoxPro reasonably compatible with dBASE IV, although it's not clear how important thss will be to users. Perhaps commercial source code developers/publishers like SBT, SourceMate, and Champion will choose to support only the command subset that will run with both products. Most dBASE IV applications should run unchanged in FoxPro, unless they rely on dBASE-only capabilities. For example, many dBASE IV programs use the MDX-type index. FoxPro does not yet support MDX, but an application that uses the FoxBASE/dBASE III approach to indexes would run with both dBASE IV and FoxPro. FoxPro has one fundamental problem in inhibiting dBASE IV compatibility. Several features added to FoxBASE+ in 1987 have been implemented somewhat differently in dBASE IV 1.0. Yet Fox wants FoxPro to support all FoxBASE+ applications while also handling dBASE IV code. So a special command, SET COMPATIBLE TO FOXPLUS

DB4, is needed to tell FoxPro which interpretation to place on an ambiguous command. LIKE(), PLAY MACRO, SELECT (), STORE (with arrays), and various WINDOW commands are affected. FoxPro differs from dBASE IV in several areas. For example, FoxPro treats most on-screen elements as objects. In dBASE IV pop-up menus are just screen images, while in FoxPro they appear in true windows. FoxPro can write something underneath a displayed window, while dBASE IV can only write on top of a window. Fox has added command options to allow a programmer to control such interactions

and to make the code act like dBASE IV. The size and position of dBASE IV windows can be specified when defined, but are fixed once displayed. FoxPro also defaults to a fixed display, but a programmer can allow interactive user control. FoxPro also exceeds dBASE IV in such areas as BROWSE, UDFs, available work areas, speed, and, we hope, reliability. Yet FoxPro leaves some things out. dBASE IV features missing include the Query By Example facility (QBE), dBASE's more sophisticated screen painter, SQL command support, transaction tracking, a report generator capable of writing actual program code, support for MDX indexes, and the useful capabilities built into the commands SET SKIP TO and LOOKUP() (already a reserved word). The company said it plans to add at least MDX, SET SKIP, and LOOKUP to FoxPro 1.1. SQL and transaction processing will be part of a future server version.

Perhaps the most noticeable divergence from dBASE IV stems from FoxPro's intimate support for a mouse. Any FoxBASE+ or dBASE program, run in FoxPro, will automatically work with a mouse. All aspects of FoxPro interactive and application-driven activity are much more fluid for those who choose to use a mouse. Beginners and poor typists are especially advised to adopt a mouse with FoxPro. Several functions are available in FoxPro to monitor mouse position and button clicks, making it possible to invent new mouse-driven routines for applications. Because mouse random access isn't always appropriate, a READ...NOMOUSE option is available when the user shouldn't be allowed to jump among GETs. Fox wants you to try a mouse so much that they'll sell you one cheap (a Dexxa Mouse for $34.95 plus $5 shipping). FoxPro supports only Microsoft compatible mice (such as Logitech, which makes Dexxa). Various other mouse brands and models may not work without a new driver.

Connectivity

FoxPro/LAN, expected about the time you read this, will reduce the need to do explicit record and file locking in several ways. In most cases, FoxPro will attempt to automatically issue a lock when needed. As in FoxBASE+/LAN GET/READ and REPLACE require that the record be locked first. If not, FoxBASE+ won't do anything but produce an error. FoxPro will first try to lock the record. A new command borrowed (and enhanced) from dBASE IV tells FoxPro what to do if the first lock attempt is unsuccessful. You can SET PROCESS AUTOMATIC retries indefinitely while displaying an "Attempting to lock, press ESC to cancel" message. If the user does cancel, an ON ERROR routine can take appropriate action. FoxPro/LAN should run quite a bit faster than FoxBASE+/LAN 2.1, which was never optimized for multiuser activities. (Fox plans to release updates to speed up the multiuser performance of FoxBASE+ and FoxBASE+/Mac soon after FoxPro/LAN appears). A READ...TIMEOUT option has been added as a way to sidestep network lock-ups when a user goes to lunch in the middle of a GET...READ.

Because dBASE IV has it (though it's been removed from version 1.1), Fox showed an early version of FoxPro with SQL at COMDEX/Fall '88. But Fulton doesn't believe anyone wants or needs local workstation SQL support. dBASE IV-like transaction processing and SQL language support will be implemented when FoxPro is interfaced to a network database server.

FoxPro 1.1

To get FoxPro 1.0 out the door, some features have been held until version 1.1. According to the company, the 64K limit on string- **pool size** will be lifted, such dBASE IV innovations as LOOKUP(), SET SKIP TO and MDX multiple index file support will be added, and the menuing style of FoxBASE+/Mac, which Fulton calls "the real FoxPro menus," will appear. The next release of FoxPro is expected to give programmers access to various internal FoxPro components such as dialog boxes and radio buttons. According to Fulton, FoxPro 1.1 will allow creation of Clipper-like EXE programs, removing the need to use FoxPro or a runtime version to execute an application. For some people, this will make FoxPro a "real" programming language (though Fulton considers it a "packaging issue"). Like Clipper, FoxPro's runtime library will be built into the EXEs it generates. To save disk space, Fox plans to make it optionally possible to reduce EXE size by specifying just one runtime library for all FoxPro EXEs on a system. EXE generation will be nice, but the most far- reaching enhancement in FoxPro 1.1 will be a documented Applications Programming Interface (API). With access to low-level FoxPro internals, developers may be able to add an endless range of capabilities. Expect a lively third-party aftermarket to develop for FoxPro 1.1 much as it has for Clipper.

FoxPro 2.0

According to Fulton, central database server support will be added in FoxPro 2.0, due sometime in 1990. Depending on demand, several server engines may be interfaces. The first announced FoxPro server will be a customized version of Novell's NetWare SQL, which Fox will call FoxServer. (Netware SQL runs as part of Novell NetWare. It appears that only the expensive NetWare 386 will be supported, not the ubiquitous Advanced NetWare 286. Smaller companies may not be willing to upgrade to a 386 server and pay an extra $5,000 to Novell just to give FoxServer its required environment.) Alternatively, the FoxPro 1.1 applications programming interface might be flexible enough to allow third-party support for various database servers, like those from Oracle, Sybase (the same one used by Ashton-Tate and Microsoft in SQL Server), and Emerald Bay. Wayne Ratliff's Emerald Bay database engine could be especially welcome because it requires nothing more than an XT-class computer on Novell or any NETBIOS-compatible network. Ratliff and Fox president David Fulton have discussed this possibility.

Nothing to fear but ourselves.

FoxPro is a technical success, approachable by non-programmers and sophisticated developers alike. For new users, learning FoxPro, even at the system menu level, may not be a trivial task. For starters, users must understand database management issues and dBASE conventions. But the reward for learning FoxPro basics is entry into the almost-unlimited world of dBASE. Like earlier forms of dBASE, once learned, FoxPro is easy to use. The ability to intermix menu choices, the dBASE/FoxPro language, and saved procedures (programs) gives FoxPro users tremendous flexibility and control. FoxPro lets you manually type whatever commands you choose, while using the menu system to have FoxPro itself type commands for more complex or obscure activities. By watching what FoxPro types into the command window, you can learn the correct syntax for every action.

Old hands experienced with FoxBASE+ or dBASE III PLUS at the dot prompt may feel constrained by FoxPro's structured interface and menus, but that's a premature judgement. It is quite possible to resize the command window to provide the same feel as the old interactive command line, and the other FoxPro facilities are unprecedented. Beside, you can always ignore the parts that don't appeal to you.

To some people, the main wildcard in FoxPro's future is Fox Software itself. Fox has credited its small size, family atmosphere, and tightly-knit, long-standing programming team for its technical accomplishments and customer satisfaction. In 1984, when dBASE III was introduced, Ashton-Tate had similar qualities. Look what success brought. Can Fox Software grow, perhaps rapidly, without losing the very qualities that made FoxPro possible? Will we like the new Fox? For Dave Fulton and his talented staff, creating FoxPro may have been the first challenge; surviving its success may be the second.

Buying FoxPro

FoxPro is priced at $795 for the single-user version, $1,095 for FoxPro/LAN, and an additional $500 for an unlimited runtime version that works in either single or multiuser mode depending on the serial number of the master copy. The company expected LAN and runtime versions to go into beta during December, and to be available in final form during late January or early February. At that same time the company will send out FoxPro 1.01, a free maintenance upgrade, to all registered users. To encourage mouse use, Fox is selling serial mice for $34.95. Upgrading from single-user FoxBASE+ to FoxPro is $195, from FoxBASE+/LAN to FoxPro/LAN, $250, and from FoxBASE+ Runtime to FoxPro Runtime, $50. All upgrades are done directly through Fox Software. It was originally announced that the upgrade from 1.0 to FoxPro 1.1, when available, would be free, but there may be a charge if 1.1 turns out to significantly expand features. FoxBASE+ will remain on the market.

Fox Software is located at 134 W. South Boundary, Perrysburg, Ohio 43551, (419) 874-0162.

John L. Hawkins is president of HawkTek Corp., a national computer and management consulting database development and VAR firm. John is also author of FoxPro Programming, to be published by Scott, Foresman and Co. and Contributing Editor of Data Based Advisor. Reach him at P.O. Box 6476, San Rafael, Calif. 94903, (415) 491-HAWK.



COMPANY NAMES: Fox Software Inc.--Products

DESCRIPTORS: Software Packages; Evaluation; DBMS
SIC CODES: 7372 Prepackaged software
TRADE NAMES: Microsoft FoxPro (Database application development software)
--evaluation
FILE SEGMENT: CD File 275
?